EXPERIMENT THREE

🌟 Intelligent Agents & Search Strategies

This file covers:

- A simple simulation of the **Vacuum Cleaner World**
- Implementation of **Greedy Best First Search**

1. Vacuum Cleaner World with Two Locations

Problem Statement:

Design a simple vacuum cleaner world consisting of only **two locations** (A and B).

i. Identify Percepts and Actions

- **Percepts**:
  - Location: A or B
  - Status: Clean or Dirty
- **Actions**:
  - Move Left
  - Move Right
  - Suck

🔧 Code Implementation

Below is a simple Python simulation of the two-location vacuum cleaner world.

```
# Vacuum Cleaner World with Two Locations

class VacuumCleanerAgent:
    def __init__(self):
        self.location = 'A'
        self.environment = {'A': 'Dirty', 'B': 'Dirty'}
        self.score = 0

    def perceive(self):
        return self.location, self.environment[self.location]

    def act(self, percept):
        location, status = percept
        if status == 'Dirty':
            self.environment[location] = 'Clean'
            self.score += 10
            action = 'Suck'
        else:
            if location == 'A':
                self.location = 'B'
            else:
                self.location = 'A'
            self.score -= 1  # movement cost
            action = f'Move to {self.location}'
        return action

    def run(self, steps=10):
        for _ in range(steps):
            percept = self.perceive()
            action = self.act(percept)
            print(f"Percept: {percept}, Action: {action}, Score: {self.score}")

# Run the agent
agent = VacuumCleanerAgent()
agent.run()
```

```
Percept: ('A', 'Dirty'), Action: Suck, Score: 10
Percept: ('A', 'Clean'), Action: Move to B, Score: 9
Percept: ('B', 'Dirty'), Action: Suck, Score: 19
Percept: ('B', 'Clean'), Action: Move to A, Score: 18
Percept: ('A', 'Clean'), Action: Move to B, Score: 17
```

```
Percept: ('B', 'Clean'), Action: Move to A, Score: 16
Percept: ('A', 'Clean'), Action: Move to B, Score: 15
Percept: ('B', 'Clean'), Action: Move to A, Score: 14
Percept: ('A', 'Clean'), Action: Move to B, Score: 13
Percept: ('B', 'Clean'), Action: Move to A, Score: 12
```

2. Greedy Best First Search Implementation

🔧 Code Implementation

Below is a simple Python simulation of the two-location vacuum cleaner world.

```python
from queue import PriorityQueue

def greedy_bfs(graph, start, goal, heuristic):
    visited = set()
    queue = PriorityQueue()
    queue.put((heuristic[start], start, [start]))

    while not queue.empty():
        (cost, current, path) = queue.get()
        if current == goal:
            return path

        visited.add(current)

        for neighbor in graph[current]:
            if neighbor not in visited:
                queue.put((heuristic[neighbor], neighbor, path + [neighbor]))

    return None

# Example Graph
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

# Heuristic values (lower is better)
heuristic = {
    'A': 5,
    'B': 4,
    'C': 3,
    'D': 999,
    'E': 1,
    'F': 0
}

path = greedy_bfs(graph, 'A', 'F', heuristic)
print("Greedy BFS Path:", path)
```

```
Greedy BFS Path: ['A', 'C', 'F']
```